

# Graphics Cards and GPU Computing

Trevor Cickovski

CS110M, 3/4/14

# What is a Graphics Card?

- A graphics card is responsible for taking image data and displaying it on your monitor



- NVIDIA GTX480 has 448 1.2 GHz GPUs

# What is a GPU?

- GPU stands for 'Graphics Processing Unit'
- Recall every computer comes with a Central Processing Unit (CPU)
  - Sole job is to run an instruction (i.e., a '+')
  - Standard is about 3.4 GHz currently
- These GPUs are thus slower, but...
  - This graphics card comes with 448 of them!
  - Modern: GTX780 has 2,304!

# What is this necessary?

- Suppose we did not have graphics cards...
- We have image data stored someplace that we need to render. Each pixel of an image contains three integers (R, G, B):

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image



Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor



# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor



# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# With a Single CPU...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# Implications

- With a single CPU, you have to process pixels sequentially
- A dual-core could do two at a time
- A quad-core, four at a time
- At best, you would get one row done at a time

# But if we had 16 GPUs...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# But if we had 16 GPUs...

R=0 G=255 B=0	R=255 G=255 B=0	etc.	

Image




Monitor

# Modern Screen Resolutions

- Of course it does not matter much:
  - For a 4x4 image
  - That we only render once
- When it will matter:
  - For a 1,440x900 image (typical screen resolution)
  - That we render everytime something on our screen changes
  - And of course, when we play movies or video games
    - Because, they are nothing but big sets of images!



# In Action...

- CPU vs. GPU Rendering Speed Comparison



- <https://www.youtube.com/watch?v=Htv26S-7YqU>

# To be aware...

- GPUs are no longer just limited to graphics rendering

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

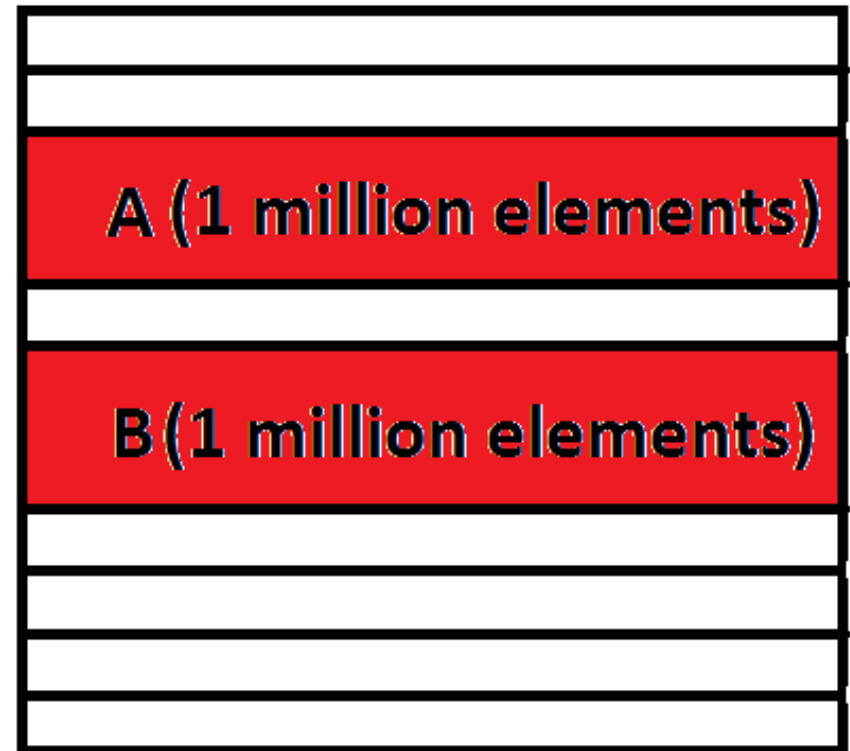
- People started realizing, if GPUs are so good at number crunching, why not apply them elsewhere?
  - 2x2 matrix addition requires 4 additions ( $2^2$ )
  - 2x2 matrix multiplication requires 4 additions ( $2^2$ ) and 8 multiplications ( $2^3$ )

# Challenges

- For so many years, GPUs were designed only to render images
  - Can only handle integers smoothly
  - Cannot interact with the user
  - Have their own, separate memory (and not much!)
- Modern research is addressing these challenges
- Let's think about  $1,000 \times 1,000$  matrices
  - Addition: 1,000,000 additions
  - Multiplication: 1,000,000 additions and 1,000,000,000 mults.

# Adding Two Matrices

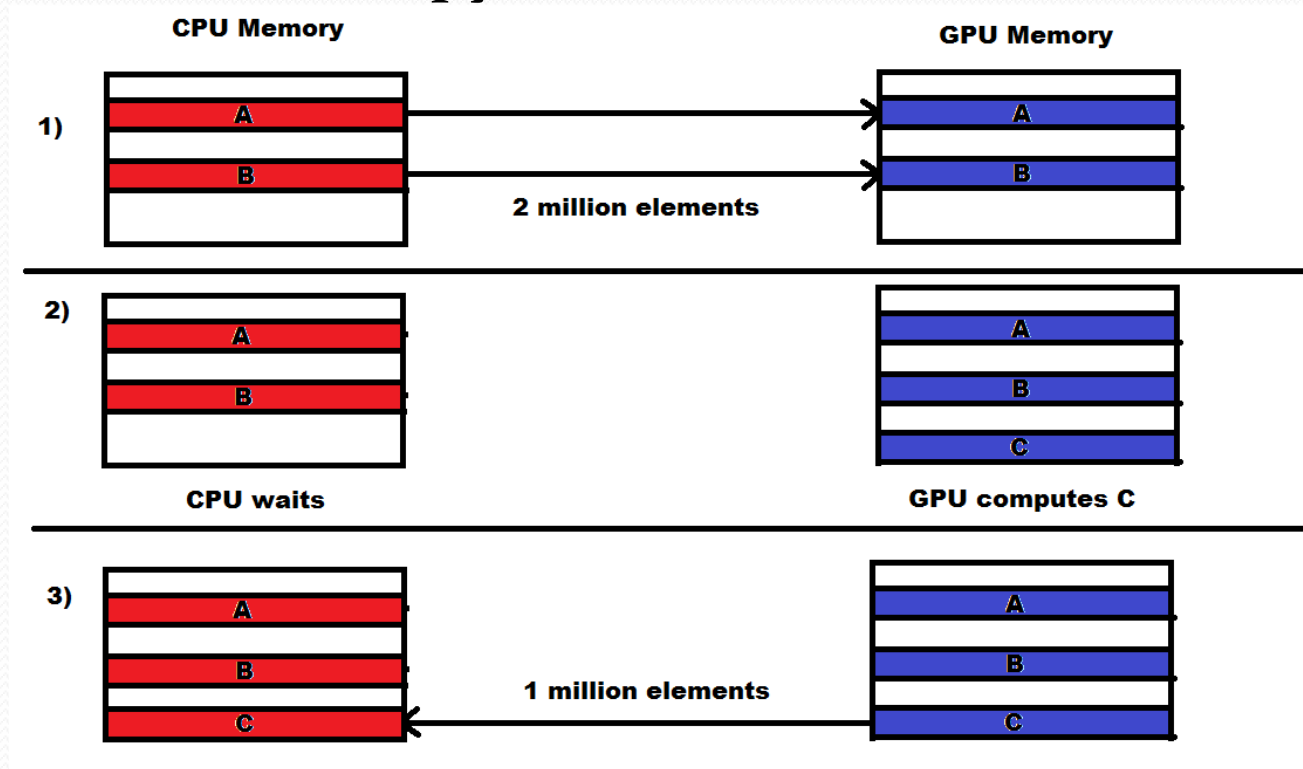
- Say A and B have 1 million elements, and we want to compute  $A + B = C$
- If we were running on the CPU...



**CPU Memory**

# Now, let's throw in the GPU...

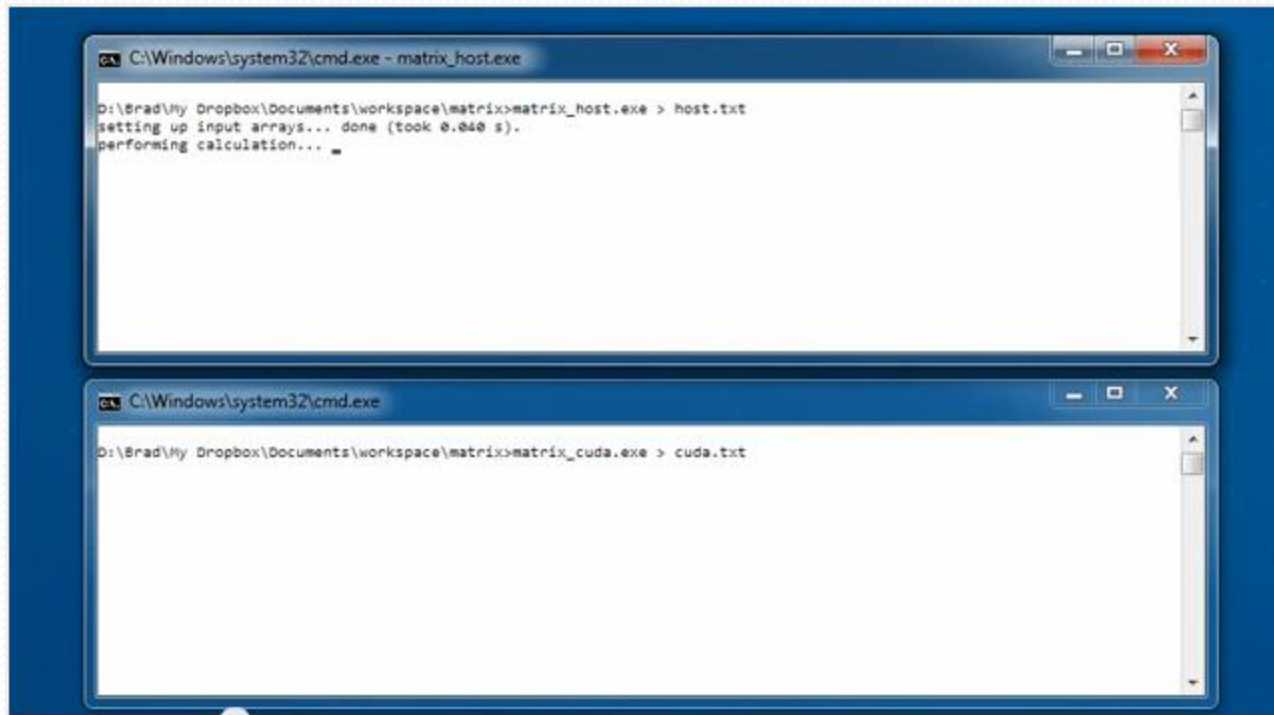
- You have to copy between memories!



- Key: Step (2) savings >>> Step (1) and (3) time

# GPU Matrix Multiplication

- Matrix Multiplication Using CUDA and GPU



The image shows two Windows command prompt windows. The top window is titled 'C:\Windows\system32\cmd.exe - matrix\_host.exe' and shows the following text: 'D:\Brad\My Dropbox\Documents\workspace\matrix>matrix\_host.exe > host.txt', 'setting up input arrays... done (took 0.040 s).', and 'performing calculation...'. The bottom window is titled 'C:\Windows\system32\cmd.exe' and shows the following text: 'D:\Brad\My Dropbox\Documents\workspace\matrix>matrix\_cuda.exe > cuda.txt'.

- Note the times for “Performing Calculation”
  - <https://www.youtube.com/watch?v=FWuGKeGEnWs>

# What CUDA Code Looks Like...

- A snapshot of Matrix Multiplication

```
// C = A * B
__global__ void matMul(int* A, int* B, int* C, int Ccols,int Acols) {
    int threadNum = threadIdx.x*1024+threadIdx.x; // Overall thread number

    if (threadNum >= Acols+Ccols) return; // If out of bounds return
    int i = threadNum / Ccols; // x coordinate
    int j = threadNum % Ccols; // y coordinate

    int sum = 0; // Holds the sum
    for (int e = 0; e < Acols; e++)
        sum = sum + A[i*Acols+e]*B[e*Ccols+j]; // Loop over the eth row of A and eth column of B

    C[i*Ccols+j] = sum; // Put the final sum in C(i, j)
}
```

- Note some common constructs to Python! 😊

# Other Applications...

- For an idea of the range of GPU applications and their speedups...
- <http://www.nvidia.com/object/gpu-applications.html>
- Note a wide range!